
Automatic Near-Body Domain Decomposition Using the Eikonal Equation

Yuanli Wang¹, Francois Guibault², and Ricardo Camarero³

¹ École Polytechnique de Montréal, C.P. 6079, Succ. Centre-ville,
Montréal (QC) H3C 3A7, Canada yuan-li.wang@polymtl.ca

² École Polytechnique de Montréal, C.P. 6079, Succ. Centre-ville,
Montréal (QC) H3C 3A7, Canada francois.guibault@polymtl.ca

³ École Polytechnique de Montréal, C.P. 6079, Succ. Centre-ville,
Montréal (QC) H3C 3A7, Canada ricardo.camarero@polymtl.ca

1 Introduction

The purpose of this research is to generate high-aspect-ratio cells for structured grids in the vicinity of boundaries for wall dominated phenomena such as viscous layers in aerodynamic applications. When such domains are discretized for complex geometries, it is important that the grid fits the boundaries well, and this becomes even more difficult to achieve with strongly curved boundaries. Domain decomposition simplifies this problem by subdividing the domain into multiple blocks and makes the task of meshing complex geometries more manageable.

There are many benefits to using a multi-block strategy. It is possible to control the orthogonality and grid quality more precisely within smaller blocks, and it allows different mesh types and generation techniques in each individual block. Also, parallel algorithms can be embedded in the mesh generation algorithm, where different blocks can be assigned to different processors, thus greatly improving the efficiency of the process.

The present paper introduces an approach to automatically decompose an arbitrary complex domain into face-matching multi-blocks emanating from the boundaries. The subdivision procedure is based on the creation of offset surfaces which closely fit their geometries instead of arbitrary planes or surfaces. This fitting is achieved through a weak solution of the *Offset Distance Function*, which is a variation of the Eikonal equation. Computing the normal directions in this manner insures that they do not cross, and the resulting propagation avoids self-intersections which arise from direct construction methods.

Thus this method can be applied on an arbitrary complex domain, i.e., a concave, or a convex shape, it may have sharp corners, or even be multi-connected. In addition, the proposed method transports the original parameterization to the propagated surface which allows rigorous matching of block faces. The geometric and topological configurations are thus well defined which allows to increase the automation level without user intervention.

This paper is arranged as follows: Section 2 gives a brief survey for multi-block strategies, and the methodology used in this work. Section 3 introduces the techniques used for the computation of the front propagation, and the proposed mathematical foundation, based on a variation of the Eikonal equation, is reviewed together with the numerical scheme used to solve this form of equation. Section 4 describes the application of this method to complex industrial configurations. Finally, Section 5 summarizes the presentation.

2 Multi-Block Generation Methodology

The numerous domain decomposition methods developed over the years were devised for entire geometric domains. The present work aims to generate viscous grids, thus we will limit the approach to generating multi-blocks around solid boundaries.

One early strategy is to use unstructured meshing methods where triangular sub-domains are generated by applying the Delaunay technique [Bergman1990, Cordova1992], and then transform the triangles into quadrilaterals by removing the appropriate edges or subdividing each triangle into three quadrilaterals that are used as blocks [Bergman1990]. Advancing-front techniques have also been used to decompose the domain directly into quadrilateral [Schonfeld1991], or coarse hexahedra cells [Kim1995] for 3D problems.

Piperni and Camarero developed a domain decomposition method [piperni2003] whose basic idea is to topologically map the original domain into a rectilinear polygon. It is then decomposed into prime rectangles which are finally mapped back to the geometric domain to generate quad (2D) or hexahedra (3D) blocks. Park and Lee suggested a hyper cube++ concept, in which complex geometry is first transformed into parametric domain, after decomposing the parametric domain, these decomposing informations are mapped back to physical space [ParkLee1998].

Guibault proposed a domain decomposition method, based on a direct propagation of the boundaries using Eqn. (2), aimed at generating multiple blocks around geometric boundary area [PHD-guibault]. Topologically, each block is equivalent as a cube template. The blocks are obtained by first generating surface grids on the boundaries to be offset, then propagating these surface grid points along their normal directions. Finally, the relative topological connectivities are constructed by sequentially connecting propagated points into edges, faces and volumes. The input and output files include both geometric information, (points, curves and surfaces), and topological information (vertices, edges, faces and volumes).

The advantage of this method is that it enforces parametric consistency between the original boundary, and its offset counterpart. All the points located on the original front are propagated along their local normal directions, thus all the points on the original front and its offset share a one-to-one parametric mapping. The weakness of this method is that self-intersection eliminating algorithms are difficult to implement and may fail for severely concave regions in 3D .

The present multi-block generation strategy is directly inspired from [PHD-guibault]. The major difference lies in the computation of the normal direction, \mathbf{n} . Instead of using the boundary's geometry directly, it is calculated through the use of a distance function ϕ field. This is obtained by the weak solution of the Offset Distance Equation, rather than using the coordinates of neighboring points.

$$\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|} \quad (1)$$

This property guarantees that local normal directions will not intersect during propagation, thus self-intersections are naturally avoided by this formulation. As no additional care is required to detect or eliminate collisions, this method can be applied on arbitrary complex domains. These can be closed or even open domains (as in through flow applications) by the addition of special boundary planes (Section 4).

3 Front Propagation Model

In dealing with surface offset or propagation, two important problems arise, for which no perfectly satisfactory solution has yet been proposed. The first problem relates to the potential self-intersection of the front as it grows from the original surface as shown in Fig. 1. *Local* self-intersection may occur during propagation when the offset distance is greater than the local curvature radius in concave regions. *Global* self-intersection, on the other hand, arises when the distance between two distinct points on the curve or surface reaches a local minimum.

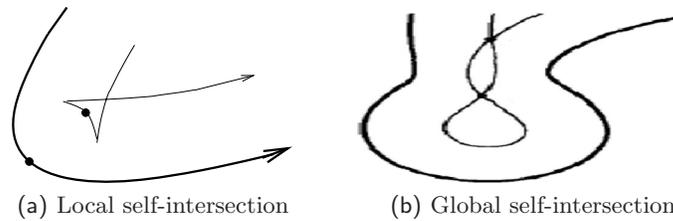


Fig. 1. Self-intersection problems

The second difficulty which is a more recent issue in offset construction is the establishment of a common connectivity between the original and offset surfaces. Current literature shows that numerous incidental applications can be facilitated if the connectivity could be transported from the original surface to its offset. These areas include transport of trim curves [kumar2003] and tool path generation [park2001], and, in the present case, semi-structured mesh generation in near wall regions.

3.1 Literature Review for Front Propagation

A variety of contributions deal with the computation of offset curves and surfaces. They can be classified in two types: *direct offset methods (DOM)*, which propagate curves or surfaces directly based on a geometric construction; *indirect offset methods (IOM)*, which cast the curve or surface offset problem into a set of partial differential equations (PDE), in which, geometric information are implicitly represented.

Direct Offset Methods (DOM)

The advantage of DOM is that the entire or partial original parameterization information can be preserved, but the self-intersections problem cannot be avoided, and extra care is required for removing self-intersections. The ability to effectively eliminate these is an important criterion in the applicability of such methods in the context of an automated procedure.

The basis for constructing the offset surface is the following equation:

$$\mathbf{x}_t = F \mathbf{n} \quad (2)$$

which relates \mathbf{x}_t , the time derivative of the geometric front position vector, to F the offset speed, a given function, and \mathbf{n} , the normal vector. The offset curves or surfaces are generated by successively solving this equation.

One representative attempt to eliminate self-intersections is the *Advancing Front Method* developed by Pirzadeh [pirzadeh1993] based on a grid-marching strategy. The solution is to simply stop the advancement of the front before self-intersections occur. Based on a similar marching idea, Sullivan [sullivan1997] presented a self-intersection detection and removal algorithm in 2D. The 3D algorithm developed by Guibault [PHD-guibault] eliminates self-intersections by first detecting tangled-loop and then re-locating the points located within this area. In the algorithm described by Glimm and al [glimm2000], a hybrid algorithm is applied to resolve self-intersections by either re-triangulating triangles after removing unphysical surfaces, or reconstructing the interface within each rectangular grid block in which crossing is detected.

Other types of techniques to eliminate self-intersections use the properties of curves and surfaces, i.e. control points, derivatives, curvature etc. Blomgren [blomgren1981], Tiller and Hanson [tiller1984], and Coquillar [coquillart1987] approached the problem by offsetting the control polygon for NURBS curves. Nachman [nachman2002] extended this idea to propagate surfaces by offsetting control points. Piegl and Tiller [piegl1999] sampled offset curves and surfaces based on bounds on the second derivatives to avoid self-intersections. In the method developed by Sun and al. [sun2004], control points are repositioned to reduce local curvature in areas where local self-intersections may occur, while the rest of the control points remain unchanged. Farouki [farouki1986] described an algorithm which first decomposes the original surfaces into parametric patches, and then uses Hermite interpolation to construct the offset surfaces.

Indirect Offset Methods (IOM)

Self-intersection problems can be avoided, but at the cost of completely losing the connectivity information stored in the original geometric front. In general, to restore a similar connectivity between the original and offset fronts is not a trivial task.

The *Level set method* developed by Sethian and Osher [osher1988], models front propagation problems as a hyperbolic differential equation.

$$\phi_t + \nabla \phi \cdot \mathbf{x}_t = 0 \quad (3)$$

in which, \mathbf{x} is the coordinate of an arbitrary point in space, ϕ is the minimum Euclidean distance from \mathbf{x} to the front to be propagated. Numerical scheme and optimization method for solving Eqn. (3) are discussed in works by Sethian [sethian1996,

sethian1999]. Kimmel [kimmel1993] applied the level set equation to offset NURBS curves and surfaces in 1993.

The significant improvement of this approach is that it intrinsically prevents self-intersections by constructing a weak solution to the offset problem. In this method, corners and cusps are naturally handled, and topological changes occur in a straightforward and rigorous manner. Complex motion, particularly those that require surface diffusion, sensitive dependence on normal directions to the interface, and sophisticated breaking and merging, can be straightforwardly implemented, with no user intervention ([malladi1996]).

In the level set method, the result of offsetting curves or surfaces are dependent on many other factors, such as the calculation of offset speed F and curvature k . Re-initialization of the level sets during calculation are required to maintain an accurate level set representation. Also, offset surfaces are extracted from the computed ϕ solution as iso-value surfaces, resulting in triangulated surface representation, which have to be re-parameterized, if the original surfaces are represented by two-dimensional parameterization (which may also be called topological connectivity if original surfaces are represented by triangles rather than parameterized surfaces). Transforming this parameterization or connectivity information into offset surfaces is a costly task. Several methods [sheffer2001, hormann2001] have been proposed to re-construct two-dimensional parameterizations based on triangulated surfaces but cannot be used to establish a one-to-one parametric relationship between the original and offset surfaces.

The advantages and deficiencies of the DOM and IOM methods are complementary. Among all of these reviewed propagating methods, the level set method prevails over other methods, and brings a significant improvement and elegant way in avoiding self-intersection problems. The present work proposes to combine the advantages of the two types of methods to build a new offset construction method that maintains parametric connectivity between the original and offset surfaces, and still avoids self-intersections through the use of a weak solution to the shortest distance problem.

3.2 Offset Computation Equation

The present work proposes the use of another type of PDE for surface offset construction, the *Offset Distance Equation*, which is a variation of the Eikonal equation:

$$\begin{cases} \nabla\phi \cdot \nabla\phi = 1 \\ \phi = 0 \quad \text{if } \mathbf{P} \in \Gamma \end{cases} \quad (4)$$

where ϕ is the minimum Euclidean distance from an arbitrary point (P) in the computational domain (Ω) to the front (Γ) to be propagated. Eqn. (4) expresses the condition for the shortest Euclidean distance from any space position to the boundary. It thus simplifies the level set mathematical model.

The ϕ function is very important in the present work: (1) It provides the basis for the calculation of the local normal directions to preserve the advantages of level set method. That means multi-block partition lines are propagated in the ϕ space, rather than in the geometric space; (2) It is also used to prevent global self-intersections. When ϕ decreases, the propagated point is approaching a boundary, and propagation is stopped to avoid collisions. (3) It can also be used as a direct stopping criterion

when the value of ϕ at the propagated point is equal or greater than the prescribed propagated distance.

Mathematical representation

Let Γ be a continuous front to be propagated, it can be either a closed or open front; \mathbf{P} be an arbitrary point in the domain of interest Ω , $\Gamma \subset \Omega$; and ϕ be the smallest Euclidean distance between \mathbf{P} and Γ . To represent the ϕ function, a point \mathbf{P}_0 is defined which meets the following conditions: (1) \mathbf{P}_0 is located on Γ ; (2) the Euclidean distance between \mathbf{P} and \mathbf{P}_0 is the minimum Euclidean distance between \mathbf{P} and Γ . Let $\mathbf{P} = (x, y, z)$, $\mathbf{P}_0 = (x_0, y_0, z_0)$, then the ϕ function can be written as

$$\phi(x, y, z) = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2} \quad (5)$$

in which, ϕ is 0 when \mathbf{P} lies on Γ . After a squared summation of the first derivatives of Eqn. (5), in a 3D Cartesian frame of reference, the distance offset equation can be written as:

$$\left(\frac{\partial\phi}{\partial x}\right)^2 + \left(\frac{\partial\phi}{\partial y}\right)^2 + \left(\frac{\partial\phi}{\partial z}\right)^2 = 1 \quad (6)$$

The problem can now be formally cast as an initial value problem using Eqn. (4) for any dimension or frame of reference.

Numerical method

The focus is now on the choice of a convergent, consistent and efficient way to solve Eqn. (4). The Fast Sweeping Scheme (FSS) developed by Zhao [zhao2004] is proposed, since it improves the computational complexity from $\Theta(N \log(N))$ in the fast marching method (FMM) developed by Sethian [sethian1999siam] to $\Theta(N)$, where N is the total number of Cartesian grid points. A brief description of the 3D numerical scheme is given to illustrate the algorithm. More details about this scheme and its validation can be found in [zhao2004, 1].

A 3D uniform Cartesian grid system is used to discretize Eqn. (6) with step sizes $\Delta x = \Delta y = \Delta z = h$. The partial derivatives $\frac{\partial\phi}{\partial x}$, $\frac{\partial\phi}{\partial y}$ and $\frac{\partial\phi}{\partial z}$ are replaced by the following Godunov upwind scheme [rouy1992],

$$\begin{aligned} \frac{\partial\phi}{\partial x} &= \frac{\phi_{i,j,k}^{t+1} - a}{h}, \\ \frac{\partial\phi}{\partial y} &= \frac{\phi_{i,j,k}^{t+1} - b}{h}, \\ \frac{\partial\phi}{\partial z} &= \frac{\phi_{i,j,k}^{t+1} - c}{h}. \end{aligned} \quad (7)$$

In the above equations, (i, j, k) are the indices for the Cartesian grid nodes in 3D, t is the sweeping time, and a, b, c are defined as

$$\begin{aligned} a &= \min(\phi_{i-1,j,k}^t, \phi_{i+1,j,k}^t) \quad i = 2, \dots, I - 1 \\ b &= \min(\phi_{i,j-1,k}^t, \phi_{i,j+1,k}^t) \quad j = 2, \dots, J - 1 \end{aligned}$$

$$c = \min(\phi_{i,j,k-1}^t, \phi_{i,j,k+1}^t) \quad k = 2, \dots, K - 1 \quad (8)$$

In Eqn. (8), I , J and K are the grid numbers in x , y and z directions, respectively. One sided-difference schemes are used for calculating a , b and c when $i = 1, \text{ or } I$, $j = 1, \text{ or } J$, and $k = 1, \text{ or } K$.

After replacing $\frac{\partial \phi}{\partial x}$, $\frac{\partial \phi}{\partial y}$, and $\frac{\partial \phi}{\partial z}$ by the expressions in Eqn. (7), and substituting into Eqn. (6) gives,

$$\left[(\phi_{i,j,k}^{t+1} - a)^+ \right]^2 + \left[(\phi_{i,j,k}^{t+1} - b)^+ \right]^2 + \left[(\phi_{i,j,k}^{t+1} - c)^+ \right]^2 = h^2 \quad (9)$$

where function $(m)^+$ is defined as:

$$(m)^+ = \begin{cases} m, & \text{when } m > 0 \\ 0, & \text{when } m \leq 0 \end{cases}.$$

After re-arranging a , b , and c into the order of $a < b < c$, $\phi_{i,j,k}^{t+1}$ can be calculated as:

$$\text{let } \phi_{temp} = a + h$$

1. if, $\phi_{temp} \leq b$, then $\phi_{i,j,k}^{t+1} = \phi_{temp}$, and it is the solution to Eqn. (6);
2. else, let $\phi_{temp} = \frac{(a+b) + \sqrt{2h^2 - (a-b)^2}}{2}$
 - a) if, $\phi_{temp} \leq c$, then $\phi_{i,j,k}^{t+1} = \phi_{temp}$, and it is the solution to Eqn. (6);
 - b) else,

$$\phi_{i,j,k}^{t+1} = \frac{(a+b+c) + \sqrt{3h^2 + 2(ab+bc+ac - a^2 - b^2 - c^2)}}{3},$$
 and it is the solution to Eqn. (6).

The new value $\phi_{i,j,k}^{new}$ at node (i, j, k) is chosen as: $\phi_{i,j,k}^{new} = \min(\phi_{i,j,k}^t, \phi_{i,j,k}^{t+1})$. The old value $\phi_{i,j,k}^t$ is always replaced by $\phi_{i,j,k}^{new}$ in the sweeping calculation process.

4 Methodology and Application

The proposed method is presented in the context of a 3D industrial geometric model, a draft tube. The characteristics are that it is a multi-connected open domain, with sharp corners and it has both concave and convex shapes. The goal is to decompose the domain into multiple hexahedral blocks emerging from the walls or boundary surfaces. The first step of the procedure is to decompose the draft-tube surface into a set of four-sided patches (see Fig. 2). These will be propagated or swept into the domain, and the original patch and the propagated patch will constitute the top and bottom faces, respectively, of a new block.

There are five essential steps in this method: (1) define a domain Ω , and discretize it into a uniform Cartesian grid; (2) calculate ϕ for each grid node using the fast sweeping algorithm; (3) calculate normal directions for the front grid nodes; (4) propagate front points along their local normal directions according to the given distance; (5) construct blocks or the relative topological connectivities around boundary area.

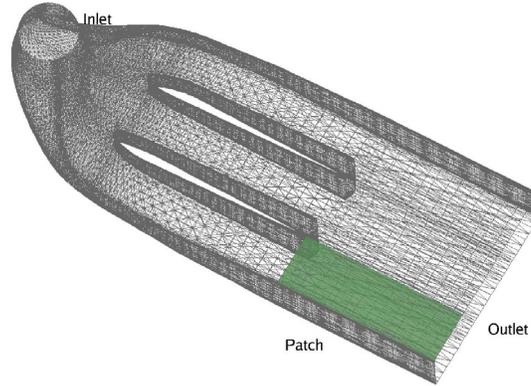


Fig. 2. Geometric model: a draft tube with 2 piers

4.1 Domain Discretization

The given geometry Γ is plunged into a domain Ω large enough to cover Γ . The domain Ω is discretized into a uniform Cartesian grid, and for each grid node, a scalar value ϕ , the minimum Euclidean distance of the Cartesian node to Γ , will be computed.

4.2 Computation of the distance field ϕ

All the grid nodes are first initialized to a large positive value. This value is chosen by the user with the only requirement is that it should be larger than the maximum possible overall distance of the computation domain. In practice, a value which is equal or greater than the size of Ω is assigned. Then two subsequent steps are applied as described below.

Initialization of Boundary-nodes

If Γ intersects an edge constructed between two grid nodes, then all the nodes belonging to the cells sharing this edge are marked as *boundary-nodes* (see Fig. 3). To enforce the boundary condition, $\phi = 0$ when $(x, y, z) \in \Gamma$, exact values are assigned to boundary-nodes. These are calculated by projecting a boundary-node onto the front segments and the minimum length of this projection line is used for initializing the ϕ value of this boundary-node. If no projection line is found for a given boundary node, then the minimum distance between this boundary-node and the node located on front segment is the exact value for this boundary-node. To accelerate boundary-node detection, the line segments in 2D (or triangles in 3D) are stored in an Alternating Digital Tree (ADT) [bonet1991].

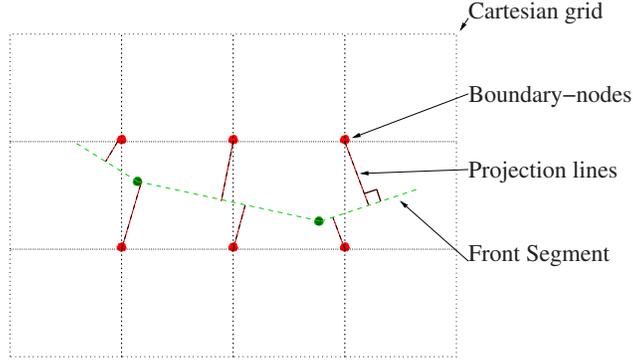


Fig. 3. Definition of Boundary-nodes

Updating the ϕ field

Gauss-Seidel iterations with alternating sweeping orderings are used to update ϕ 's. In 2D, it takes 4 sweeps to get a converged solution, each sweep is carried out along the diagonal direction of the Cartesian grid. After each sweep, one-quadrant data located in the destination area of the sweeping direction is updated. The same sweeping procedure is applied for the 3D calculation. Since there are eight vertices in a 3D Cartesian grid, it takes 8 sweeps to update the data located in 8 octants. Algorithm 1 illustrates this ϕ updating procedure for one sweep, all the other seven sweeps are similar.

Algorithm 1: updating ϕ (one sweep procedure)

```

for  $k = 0, K$  do
  for  $j = 0, J$  do
    for  $i = 0, I$  do
      calculate  $a, b,$  and  $c$ 
      calculate  $\phi_{i,j,k}^{t+1}$ 
      calculate  $\phi_{i,j,k}^{new}$ 
    end for
  end for
end for

```

4.3 Calculation of the normal directions

At each grid node, ϕ_x , ϕ_y and ϕ_z are calculated using a centered difference scheme:

$$\phi_x = \frac{\phi_{i+1,j,k} - \phi_{i-1,j,k}}{2\Delta x}, \quad i = 2, \dots, I-1 \quad (10)$$

$$\phi_y = \frac{\phi_{i,j+1,k} - \phi_{i,j-1,k}}{2\Delta y}, \quad j = 2, \dots, J-1 \quad (11)$$

$$\phi_z = \frac{\phi_{i,j,k+1} - \phi_{i,j,k-1}}{2\Delta z}, \quad k = 2, \dots, K-1 \quad (12)$$

and a one-sided difference scheme is used for boundary nodes, when $i = 1$ or $I, j = 1$ or J and $k = 1$ or K .

At each grid node on the front, (x_P, y_P, z_P) , the values of ϕ_{x_P} , ϕ_{y_P} and ϕ_{z_P} are interpolated using the values at eight neighboring grid nodes $(\phi_{x_i}, \phi_{y_i}, \phi_{z_i})$, $i = 1, \dots, 8$, as shown in Fig. 4. The normal directions at front points can be calculated directly using Eqn. (1).

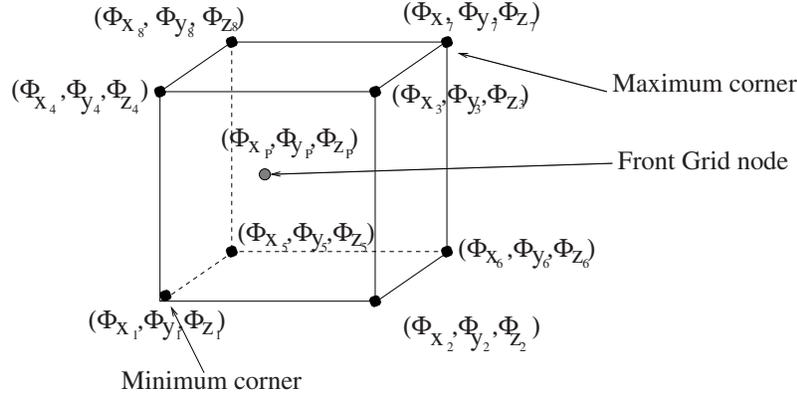


Fig. 4. Calculation of ϕ_{x_P} , ϕ_{y_P} and ϕ_{z_P}

For an open domain, in addition to the boundary surfaces, it is necessary to specify the boundary curves which bound the boundary surface. For example, the inlet circle and outlet rectangle in Fig. 5 are such boundary curves. These lie on special boundary planes which are added for the purpose of correctly closing the domain from a topological point of view, as well as providing the support for the propagation of these boundary curves.

Within each such surface, the boundary curve propagation problem is transformed into a 2D curve propagation problem, and the algorithm described in this section is applied directly. In the present application, it is required that normal directions should lie on such special boundary planes, i.e. this algorithm cannot be used when the boundary curves are general space curves.

4.4 Point Propagation

Replacing \mathbf{x}_t by a finite difference scheme, and letting $F = 1$, Eqn. (2) can be rewritten as

$$\mathbf{x}^{n+1} = \mathbf{x}^n \pm \frac{\nabla\phi}{|\nabla\phi|} dt \tag{13}$$

in which n is the propagation time, and dt is the time step. The term $\nabla\phi/|\nabla\phi|$ is calculated by a forth-order Runge-Kutta method [hoffman1992] and can be viewed as a unit propagation speed in the normal direction; positive for an outward propagation, and negative for an inward propagation.

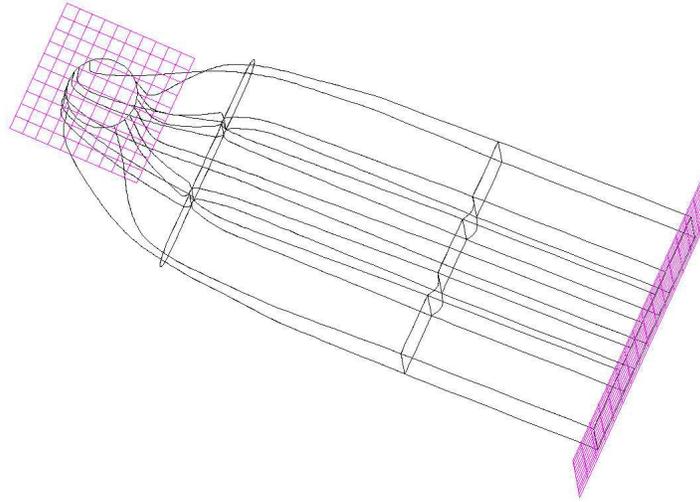
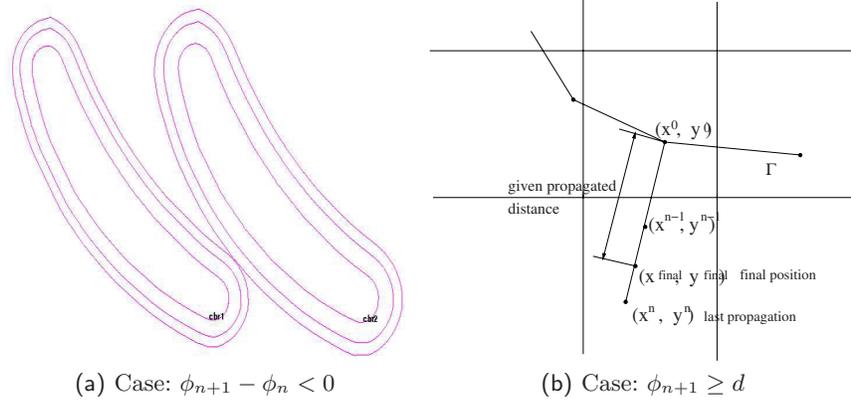


Fig. 5. Boundary curves and Special boundary planes

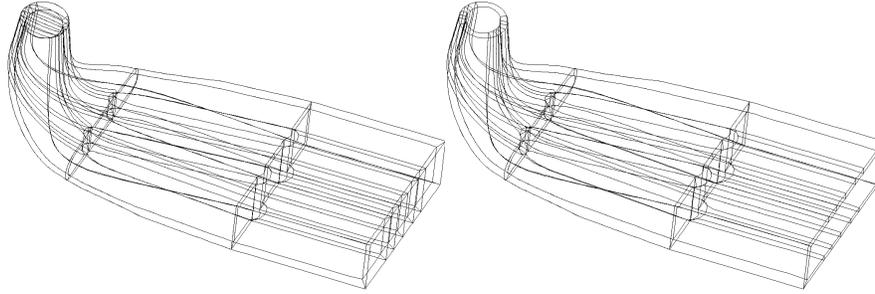


(a) Case: $\phi_{n+1} - \phi_n < 0$

(b) Case: $\phi_{n+1} \geq d$

Fig. 6. Propagation stopping criteria

For each new propagated point, there are two stopping criteria. When one of them is satisfied, the propagation is stopped (see Fig. 6): (1) If $\phi_{n+1} - \phi_n < 0$, or in other words, if ϕ starts to decrease; (2) If ϕ is equal to or greater than the specified propagation distance, d . When the value of ϕ at the final position \mathbf{x}^{n+1} is greater than the given propagation distance, then linear interpolation is used to get the exact position.



(a) The resulting blocks - closed domain (b) The resulting blocks - open domain

Fig. 7. Result comparisons (overall views)

4.5 Construction of topological connectivities

All the propagated points are sequentially connected according to their original connectivities, and corresponding points on the original and propagated geometries are used to construct blocks. Detailed descriptions about topological connectivities can be found in [PHD-guibault].

Fig. 7 illustrates the difference of the resulting blocks for a closed and open domain. In Fig. 7(a), inlet and outlet surfaces are added to Fig. 2, in this case, the original geometry can be viewed as a closed domain. Fig. 7(b) is the resulting blocks generated by directly propagating original geometry, in this case, the original domain can be viewed as an open domain, thus special boundaries are added at inlet and outlet parts. Fig. 8 is the enlarged comparisons at inlet and outlet parts. From Fig 8(a) to 8(d), we can see that inlet and outlet surfaces are propagated and blocks are generated when the original geometry is dealt as a closed domain; and boundary curves are propagated along the inlet and outlet planes when original geometry is an open domain.

Fig. 9 is the overall view of the resulting mesh, in which a structured mesh is generated within each individual block. Fig. 10 is the enlarged view for the results generated around a pier and in a sharp corner, respectively. Fig. 11 is an enlarged view of the propagating lines at a sharp corner. Theoretically, the front does not intersect itself. However, marched elements can eventually collapse causing the volume of some of the resulting elements to vanish. Elliptical smoothing of the mesh in each block could be used to attenuate this problem.

5 Conclusion

In this paper, we have presented a new approach to construct structured and semi-structured meshes near solid boundaries, based on domain decomposition. The proposed decomposition approach first proceeds by explicitly constructing an offset surface at the boundary, which is then used to topologically subdivide the domain into simple regions that are meshed independently. A key contribution of this approach

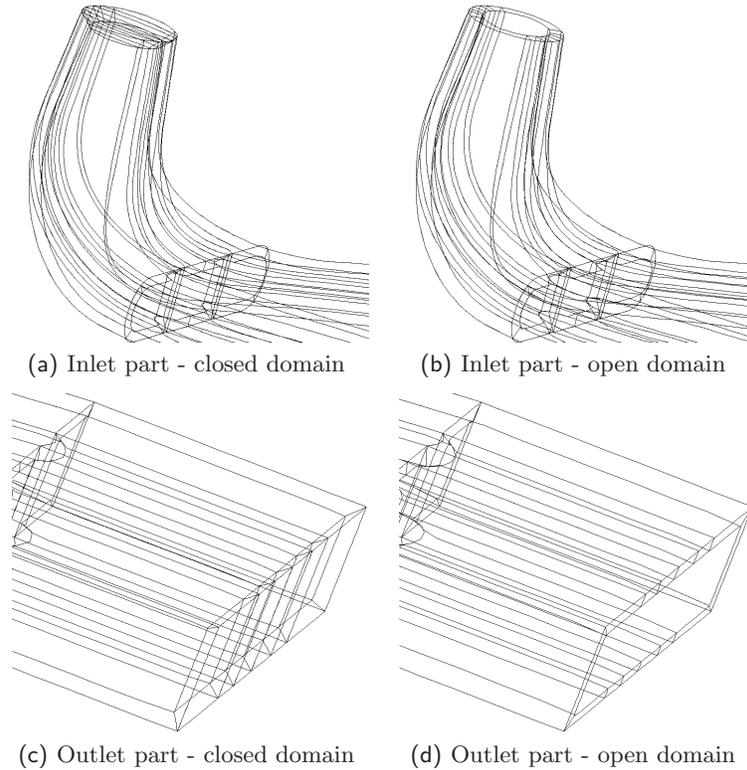


Fig. 8. Comparisons between open and closed domains (inlet and outlet part)

lies in the method used to construct the boundary offset surfaces and construct the structured mesh in each block. A mixed method for the solution of the offset distance equation is used, that alleviates local and global self-intersection problems during offset surface construction, while allowing to maintain a parametric relationship between the original surfaces and their corresponding offset.

This method can be applied indistinctly to continuous surfaces exported from CAD systems (e.g. NURBS) and to discrete polygonal versions of domain boundaries. The front can be an arbitrarily complex multi-connected domain. Topological connectivities are easily and naturally handled after propagation without user intervention, and the block faces exactly match each other. Local self-intersections are naturally avoided using the weak solution to the Eikonal equation, and global self-intersections can be avoided based on a simple detection criterion involving the ϕ field.

Permeable and internal surfaces can naturally be used to constrain the volume decomposition process through the introduction of special boundary planes. The current implementation only allows to treat planar fictitious boundaries, but there is no theoretical limit that prevents the extension to surfaces of arbitrary shape.

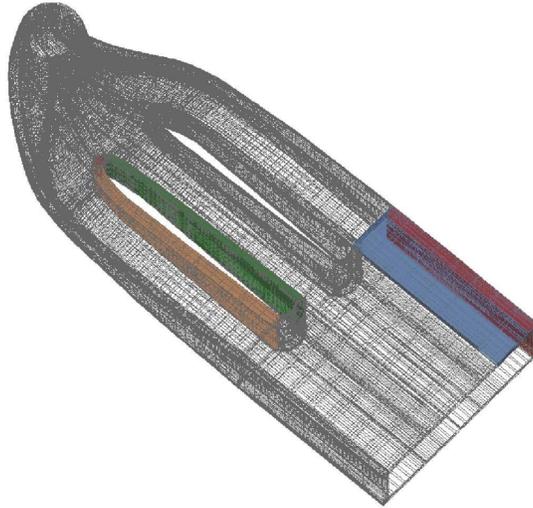


Fig. 9. Resulting mesh - overall view

6 Acknowledgments

The authors would like to express their appreciation to GE Canada (Hydro) and the National Science and Engineering Research Council of Canada (NSERC) for their support. The authors would also like to thank Ko-Foa Tchou for the valuable discussions and Mrs. Ying Zhang for supplying the 3D test cases.

- [Bergman1990] M. Bergman, "Development of numerical techniques for inviscid hypersonic flows around re-entry vehicles," *PhD thesis, INP Toulouse*, 1990.
- [Cordova1992] J. Cordova, "Towards a theory of automated elliptic mesh generation," *In NASA Workshop on Software Systems for Surface Modeling and Grid Generation, NASA Conference Publication, n 3143, p231*, 1992.
- [Schonfeld1991] T. Schonfeld and P. Weinerfelt, "The automatic generation of quadrilateral multi-block grids by the advancing front technique," *In Numerical Grid Generation in Computational Fluid Dynamics, A.S. Arcilla, J. Hauser, P.R. Eiseman, and J.F. Thompson (Eds.), p743, Proceedings of the 3rd International Grid Conference, North Holland, Barcelona, Spain, June*, 1991.
- [Kim1995] B. Kim and S. Eberhardt, "Automatic multi-block grid generation for high-lift configuration wings," *Proceedings of the Surface Modeling, Grid Generation and Related Issues in Computational Fluid Dynamics Workshop, NASA Conference Publication 3291, p. 671, NASA Lewis Research Center, Cleveland, OH, May*, 1995.
- [piperni2003] P. Piperni and R. Camarero, "A fundamental solution to the problem of domain decomposition in structured grid generation," *AIAA 2003-0951, 41st Aerospace Sciences Meeting & Exhibit, 6-9 January*, 2003.

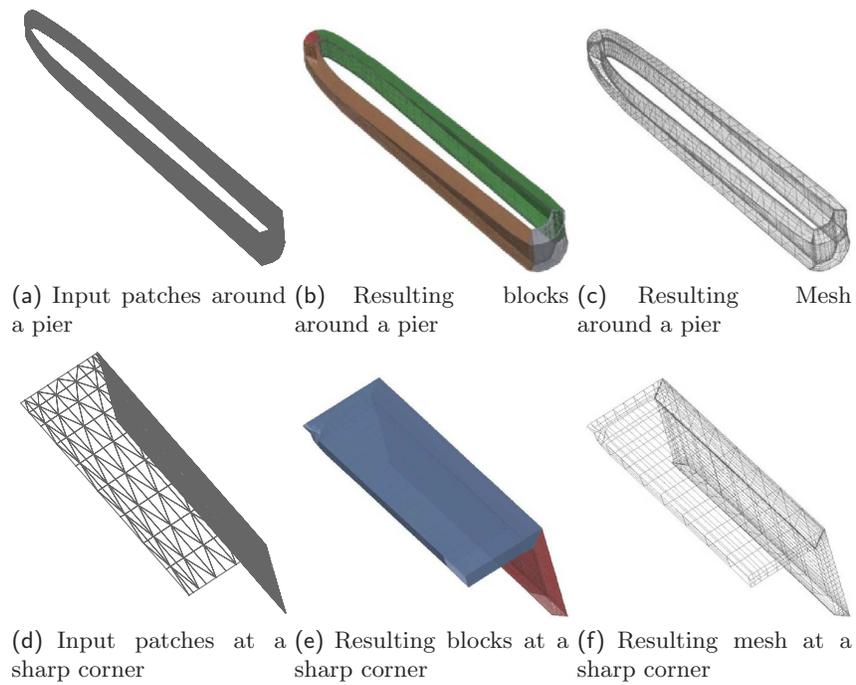


Fig. 10. Resulting mesh - partial views

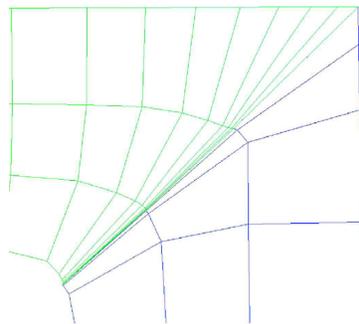


Fig. 11. Propagation behavior at a sharp corner

[ParkLee1998] S. Park and K. Lee, "Automatic multiblock decomposition using hypercube++ for grid generation," *Computers and Fluids*, 27,4,May, p509-528, 1998.

[PHD-guibault] F. Guibault, *Un mailleur hybride structuré/non structuré en trois dimensions*. PhD thesis, École Polytechnique de Montréal, 1998.

- [kumar2003] R. G. V. V. Kumar, K. G. Shastry, and B. G. Prakash, "Computing offsets of trimmed nurbs surfaces," *Comp.-Aided Design*, vol. 35, no. 5, pp. 411–420, 2003.
- [park2001] S. Park and B. Choi, "Uncut free pocketing tool-paths generation using pair-wise offset algorithm," *Comp.-Aided Design*, vol. 33, no. 10, pp. 739–746, 2001.
- [pirzadeh1993] S. Pirzadeh, "Unstructured viscous grid generation by advancing-layers method," in *AIAA 11th Applied Aerodynamics Conference*, no. AIAA-93-3453, (Monterey, CA), pp. 420–434, Aug. August 9–11, 1993.
- [sullivan1997] J. Sullivan and J. Zhang, "Adaptive mesh generation using a normal offsetting technique," *Finite Elements in Analysis and Design*, vol. 25, pp. 275–295, 1997.
- [glimm2000] J. Glimm, J. Grove, X. Li, and D. Tan, "Robust computational algorithms for dynamic interface tracking in three dimensions," *SIAM J. Sci. Comp.*, vol. 21, no. 6, pp. 2240–2256, 2000.
- [blomgren1981] R. Blomgren, "B-spline curves, boeing document, class notes, b-7150-bb-wp-2811d-4412," 1981.
- [tiller1984] W. Tiller and E. Hanson, "Offsets of two-dimensional profiles," *IEEE Computer Graphics and Applications*, vol. 4, no. 9, pp. 36–46, 1984.
- [coquillart1987] S. Coquillart, "Computing offsets of b-spline curves," *Comp.-Aided Design*, vol. 19, no. 6, pp. 305–309, 1987.
- [nachman2002] A. Kulczycka and L. Nachman, "Qualitative and quantitative comparisons of b-spline offset surface approximation methods," *Comp.-Aided Design*, vol. 34, pp. 19–26, Jan. 2002.
- [piegl1999] L. A. Piegl and W. Tiller, "Computing offsets of nurbs curves and surfaces," *Comp.-Aided Design*, vol. 31, pp. 147–156, Feb. 1999.
- [sun2004] Y. F. Sun, A. Y. C. M. Nee, and K. S. Lee, "Modifying free-formed nurbs curves and surfaces for offsetting without local self-intersection," *Comp.-Aided Design*, vol. 36, no. 12, pp. 1161–1169, 2004.
- [farouki1986] R. T. Farouki, "The approximation of non-degenerate offset surfaces," *Comp.-Aided Geom. Design*, vol. 3, no. 1, pp. 15–43, 1986.
- [osher1988] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations," *J. Comp. Phys.*, vol. 79, pp. 12–49, 1988.
- [sethian1996] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 93, no. 4, pp. 1591–1595, 1996.
- [sethian1999] J. A. Sethian, *Level set methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1999.
- [kimmel1993] R. Kimmel and A. M. Bruckstein, "Shape offsets via level sets," *Comp.-Aided Design*, vol. 25, no. 3, pp. 154–162, 1993.
- [malladi1996] R. Malladi and J. A. Sethian, "Level set and fast marching methods in image processing and computer vision," *IEEE International Conference on Image Processing*, vol. 1, pp. 489–492, 1996.
- [sheffer2001] A. Sheffer and E. de Sturler, "Parameterization of faceted surfaces for meshing using angle based flattening," *Engineering with Computers*, vol. 17, no. 3, pp. 326–337, 2001.

- [hormann2001] K. Hormann, *Theory and Applications of Parameterizing Triangulations*. PhD thesis, Department of Computer Science, University of Erlangen, Nov. 2001.
- [zhao2004] H. K. Zhao, “A fast sweeping method for eikonal equations,” *Mathematics of Computation*, vol. 74, no. 250, pp. 603–627, 2005.
- [sethian1999siam] J. A. Sethian, “Fast marching methods,” *SIAM Review*, vol. 41, no. 2, pp. 199–235, 1999.
- [1] Y.-L. Wang, F. Guibault, R. Camarero, and K.-F. Tchon, “A parametrization transporting surface offset construction method based on the eikonal equation,” in *17th AIAA Computational Fluid Dynamics Conference*, 6 – 9 June 2005.
- [rouy1992] E. Rouy and A. Tourin, “A viscosity solution approach to shape-from-shading,” *SIAM Journal on Numerical Analysis*, vol. 29, pp. 867–884, 1992.
- [bonet1991] J. Bonet and J. Peraire, “An alternating digital tree (ADT) algorithm for 3D geometric searching and intersection problems,” *Int. J. Numer. Meth. Engng*, vol. 31, pp. 1–17, 1991.
- [hoffman1992] J. Hoffman, *Numerical Methods for Engineers and Scientists*. McGraw-Hill Inc., 1992.

